Enhancing network security based on anomaly detection using deep learning for intelligent IDS/IPS systems

Hà Trọng Thắng

Faculty of Informatics, East Asia University of Technology

Email: thanght@eaut.edu.vn

Abstract

Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) are critical security solutions designed to monitor and manage network activities in order to detect and respond to attacks or abnormal behaviors. An IDS primarily monitors network traffic and generates alerts upon detecting suspicious activities, whereas an IPS can proactively respond to threats by automatically blocking packets, terminating connections, disabling accounts, or isolating devices from the network.

However, traditional IDS/IPS systems are limited in their ability to detect complex anomalous patterns in network traffic and often respond ineffectively in dynamic and heterogeneous network environments.

This research proposes an intelligent IDS/IPS framework for LAN environments, integrating a Deep Learning-based Autoencoder to perform anomaly detection on network traffic features. As an unsupervised learning approach, it enables the identification of anomalies without requiring pre-labeled attack data, while an automated alert mechanism and real-time response capability enhance the effectiveness of threat detection, monitoring, and prevention in a more proactive and efficient manner.

Keywords: *Autoencoder, Deep Learning, network monitoring, IDS/IPS, anomaly detection*

1. INTRODUCTION

The rapid development of information technology has led to the emergence of a diverse ecosystem of interconnected devices, offering exceptional convenience to users. Within this context, computer networks play a foundational role, serving as critical infrastructure for global connectivity, data transmission, and information exchange.

The exponential growth in the number of connected devices and the volume of transmitted data has spurred continuous research in computer networking - particularly in areas related to security and information protection. Today, computer networks are not merely tools for communication, but have become indispensable components across multiple domains such as business, education, healthcare, and media.

In Vietnam, cybersecurity has experienced several notable developments in recent years:

+ Number of cyberattacks: In 2024, more than 659,000 cyberattacks were recorded, with 46.15% of organizations and enterprises having experienced at least one incident [1].

+ Growing sophistication of threats: Cyberattacks have become increasingly sophisticated and complex, highlighting the urgent need to enhance the security of internal networks, including LANs [2]. Raising awareness and investing in robust cybersecurity measures are crucial for data protection, ensuring system stability, and maintaining uninterrupted operations of organizations and businesses.

Therefore, this research focuses on designing an intelligent, LAN-based AI-IDS/IPS system that leverages anomaly detection to enhance the security of internal networks.

2. RELATED WORK

Intrusion Detection and Prevention Systems (IDS/IPS) play a critical role in monitoring network traffic, analyzing behavior, and protecting systems against security threats. IDS (Intrusion Detection System) is responsible for detecting abnormal or unauthorized behaviors, while IPS (Intrusion Prevention System) actively prevents such activities [3]. However, in increasingly complex network environments and with the constant evolution of attack techniques, traditional IDS/IPS systems still exhibit significant limitations [4].

Specifically, traditional IDS only performs monitoring, detection, and alerting when abnormal signs are observed, while IPS is designed to automatically prevent attacks by dropping packets, resetting connections, or blocking IP addresses. Nevertheless, the effectiveness of IPS still largely depends on accurate detection based on predefined patterns or known signatures [5].

Furthermore, high rates of false positives and false negatives are common issues for both IDS and IPS. False positives consume system resources and affect operational efficiency, while false negatives risk overlooking actual attacks. Properly adjusting detection thresholds in traditional systems remains a major challenge, particularly in environments where traffic patterns and user behavior are continuously changing [6].

Additionally, signature-based IDS/IPS systems rely heavily on databases of known attack patterns. This dependency makes it difficult to detect novel (zero-day) attacks, sophisticated evasion techniques, or obfuscation strategies. These systems are not designed to learn from new data or adapt to emerging threats [7].

Lastly, traditional IDS/IPS systems mainly focus on detecting unauthorized access from external sources, while their ability to identify anomalies within internal networks remains limited [8].

In light of these limitations, this study proposes an integrated solution that combines Artificial Intelligence (AI) and Deep Learning to enhance control capabilities and improve decision-making in threat detection and prevention [9].

3. METHODOLOGY

To effectively enhance cybersecurity, this study proposes an intelligent IDS/IPS system improved by integrating a Deep Learning-based Autoencoder model - an unsupervised learning technique widely used in anomaly detection - for identifying deviations in network traffic [10]. The system is capable of learning normal network behavior and detecting anomalies through reconstruction error when processing connection sessions. Upon identifying an anomaly, the system triggers an automated alert mechanism via a real-time dashboard or initiates timely responses such as blocking packets from suspicious devices through Telnet communication with network equipment [11].

The combination of deep learning capabilities and real-time response enables the system to proactively detect threats at an early stage, minimize potential risks, and enhance the protection level of the internal network [12].

3.1. AI-enhanced Intelligent IDS/IPS System

3.1.1. Overall System Architecture

The operational model of the AI-based IDS/IPS system includes key components such as network data collection, data preprocessing, model training, anomaly detection, and real-time response [13].

a) Network data collection

In this research, the *scan_system* module was developed to automate data collection from the local network by leveraging Nmap for device scanning and Scapy for traffic analysis. The information collected includes the IP address, MAC address, number of packets, data volume, connection duration, and device status. This data is then stored in the *network_device_data.csv* file, which serves both real-time analysis and as input for training the Autoencoder deep learning model.

The structure of the *network_device_data.csv* file is illustrated in ***Table 1***. Data collection is typically performed

over several minutes to several hours to ensure that a sufficient quantity of data is obtained.

Table 1. Field descriptions in the *network_device_data.csv* file

Field name	Description
ip_address	IP address of the device
mac_address	Physical MAC address of the device
packets	Total number of packets sent/received during the connection session
bytes	Total amount of transmitted data (in bytes)
duration	Duration of the connection session (since first detection)
connection_time	Timestamp when the system scanned and recorded the device information
status	Status of the device (e.g., *Active*)

b) Data preprocessing and feature extraction

The collected data are preprocessed by normalizing numerical features such as *packets*, *bytes*, and *duration* using the MinMaxScaler technique, which transforms the raw values into a standard range [0, 1] according to Eq. (1):

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad (1)$$

where,

- x is the original value, and $x_{\min}$, $x_{\max}$ are the minimum and maximum values of the feature in the dataset.

- x': The normalized value of the feature, scaled to the range [0, 1]

Categorical features, such as *status*, are encoded using One-hot Encoding. As a result, the dataset is transformed into numerical vectors $x \in R^n$, where n is the total number of features after encoding. These vectors are then fed into the Autoencoder deep learning model for training. This preprocessing step helps reduce noise and improve the model's accuracy.

c) Training and anomaly detection using Autoencoder

The Autoencoder model used in this research consists of three main layers: an input layer, two hidden layers (encoder/decoder), and an output layer. It is trained to reconstruct normal network traffic data, with reconstruction errors used to detect anomalies during actual operation.

Let $x \in R^n$ denote the input feature vector. The encoder compresses the input into a lower-dimensional latent representation **z**:

$$\mathbf{z} = f_{enc}(x) = \sigma(W_e x + b_e) \quad (2)$$

where,

- f_{enc}: The encoder function that maps **x** to **z**

- W_e: The encoder weight matrix, learned during training

- b_e: The bias vector of the encoder layer

- $\sigma(.)$: A nonlinear activation function (ReLU) that models nonlinear relationships between inputs and the latent representation

The decoder then reconstructs the input from this latent vector:

$$\hat{x} = f_{dec}(\mathbf{z}) = \sigma(W_d \mathbf{z} + b_d) \quad (3)$$

where,

- $\hat{x}$: The reconstructed output vector

- f_{dec}: The decoder function that maps **z** back to the input space

- W_d: Weight matrix of the decoder

- b_d: The bias vector of the decoder layer

The network is trained to minimize the Mean Squared Error (MSE) between the input x and the reconstruction $\hat{x}$:

$$\mathcal{L}_{MSE} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{x}_i)^2 \quad (4)$$

where,

- $\mathcal{L}_{MSE}$: The Mean Squared Error loss value, measuring the average squared difference between input and reconstructed output

- n: The number of elements (dimensions) in the feature vector

- x_i: The i-th component of the original input vector

- $\hat{x}_i$: The i-th component of the reconstructed output vector

After training, the model is deployed in real-time monitoring. During inference, any input sample that yields a reconstruction error greater than a predefined threshold θ is flagged as an anomaly: If $\mathcal{L}_{MSE}(x, \hat{x}) > \theta \Rightarrow$ Anomaly Detected

The threshold θ is empirically determined based on the reconstruction error distribution of the normal training data.

❖ **The model architecture includes the following components**

- Input layer: This layer consists of 4 neurons corresponding to the normalized input features derived from real-world network data, including: *packets, bytes, duration*, and *status*. Among these, the quantitative features (*packets, bytes*, and *duration*) are normalized using the MinMaxScaler technique, whereas the categorical feature (*status*) is encoded using one-hot encoding.

- Encoder: The encoder comprises two consecutive hidden layers with 6 and 3 neurons, respectively, each utilizing the ReLU activation function. This component is responsible for reducing the input dimensionality and extracting essential features into a lower-dimensional latent representation.

- Latent space: Located between the encoder and the decoder, this layer contains 3 neurons that represent the compressed representation of device characteristics in a low-dimensional space.

- Decoder: The decoder mirrors the encoder, consisting of two layers with 3 and 6 neurons, respectively, also employing ReLU activation. This stage reconstructs the original input data from the compressed latent representation.

- Output layer: This layer contains 4 neurons - equal to the input layer - and uses a linear activation function. Its objective is to produce reconstructed outputs that closely approximate the original inputs. The reconstruction error (difference between input and output) is used as the basis for anomaly detection.

- Training configuration: The model is trained over 100 epochs, with a batch size of 16, and a learning rate of 0.001, using the Adam optimizer and the Mean Squared Error (MSE) loss function.

- Mathematical description: Mathematically, the model learns a mapping function $f: x \rightarrow \hat{x}$ such that the reconstruction error $\mathcal{L}(x, \hat{x}) = ||x - \hat{x}||^2$ is minimized for normal input data.

- Anomaly detection mechanism: During the inference phase, if a data sample x produces a reconstruction error exceeding a predefined threshold θ, it is flagged as anomalous. The threshold θ is determined based on the distribution of reconstruction errors on the normal training data using a percentile-based method, as shown in Eq. (5):

$$\theta = \text{percentile}(\mathcal{L}(x, \hat{x}), 95) \quad (5)$$

where the function percentile(·, 95) returns the 95th percentile of the error distribution on normal data, and different threshold values may significantly affect the trade-off between false positives and false negatives.

❖ Operating mechanism

An overview of the anomaly detection model based on the Autoencoder is shown in ***Figure 1***. The Autoencoder, a type of deep neural network, is capable of learning to encode and decode input data such that the reconstructed output closely matches the original input. For anomaly detection, the Autoencoder is trained in an unsupervised fashion exclusively on normal data. When it encounters anomalous data, the reconstruction error tends to be high, as the model is not familiar with the patterns present in abnormal samples [14].

Specifically, the model is trained on sessions of normal network connections to learn a compressed representation of their features. During operation, new connection sessions are input into the model to calculate their reconstruction error; if this error exceeds a predefined threshold, the session is flagged as anomalous. The system is architected with two independent modules: *train_model* and *anomaly_detector*. This modular design allows the model to be updated periodically, flexibly, and without causing interruptions to the operational system.

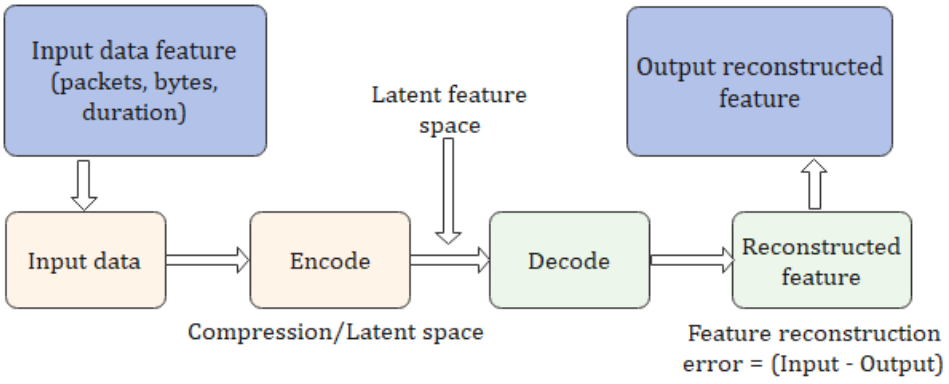


Figure 1. Autoencoder-based anomaly detection model

d) Alerting and automated response

When an anomaly is detected, the AI-IDS/IPS system automatically logs the event, generates an alert on the web dashboard, and can proactively block related traffic by interfacing with network devices via Telnet, thereby minimizing potential risks.

This mechanism ensures real-time anomaly detection with low latency, while effectively reducing the false positive rate through the use of a deep learning Autoencoder model implemented with the TensorFlow library [15].

3.1.2. Operational workflow of the AI-IDS/IPS system

The operational workflow of the AI-IDS/IPS system consists of several stages, including network data collection, data preprocessing, model training, anomaly detection, and real-time response. This process is illustrated in ***Figure 2***.

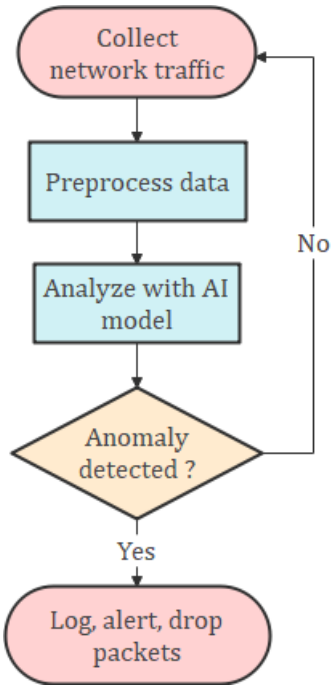


Figure 2. Operational workflow of the AI-IDS/IPS system

3.2. Deployment of the LAN model

In this research, the network model is simulated using GNS3 (Figure 3), where the AI-IDS/IPS server functions as the monitoring system. The Kali-Linux and Ubuntu-Desktop machines are used to perform various experimental attacks. Tools like *Nmap* and *hping3* are commonly used to simulate different types of network attacks, such as TCP SYN Flood and ICMP Flood, for security testing purposes.

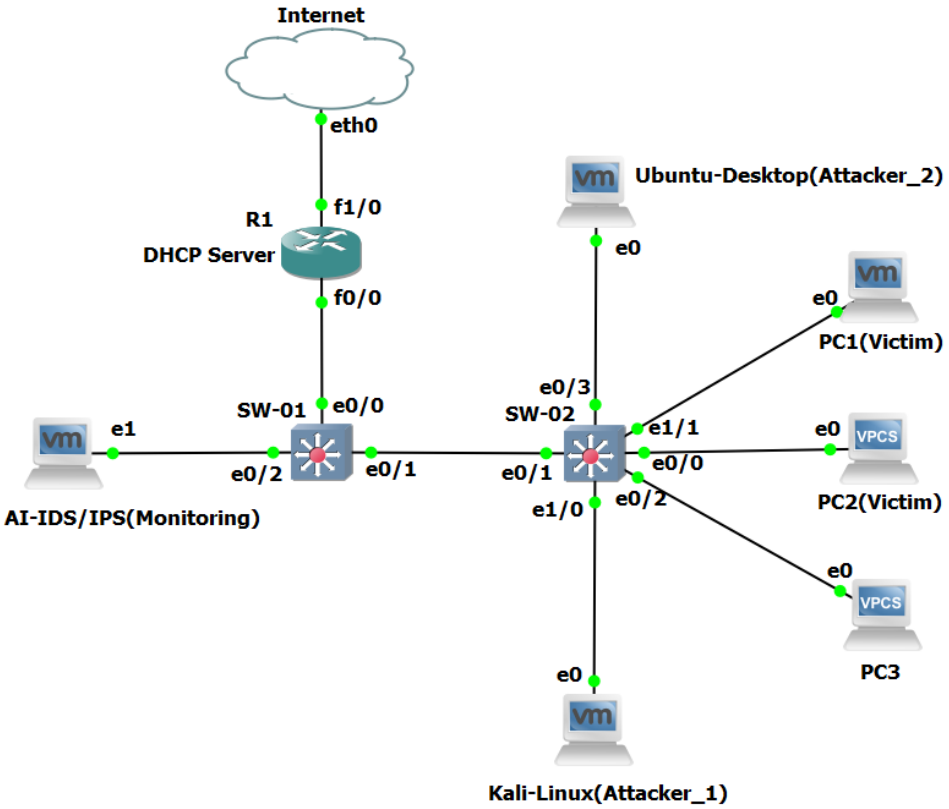


Figure 3. LAN network system model

❖ **LAN network model**

The LAN network model consists of the following components:

- Router (R1): Configured as a DHCP server with Internet access.
- Switches (SW-01, SW-02): Configured with SPAN (Switched Port Analyzer) to monitor and analyze network traffic by mirroring data from source ports to a destination port connected to the monitoring server (AI-IDS/IPS).
- Monitoring Server (AI-IDS/IPS) - Ubuntu Server: Performs network traffic monitoring, anomaly detection, and alert generation.
- Clients (PC1, PC2): Target machines used in security testing scenarios.
- Attacker_1 (Kali-Linux): Simulates different types of attacks.
- Attacker_2 (Ubuntu-Desktop): Simulates additional attack scenarios.

❖ **System operation workflow**

In the LAN network model, the system comprises a Router (R1) configured as a DHCP server, providing IP address allocation and Internet access. Two switches (SW-01 and SW-02)

operate in SPAN (Switched Port Analyzer) mode, mirroring network traffic to the AI-IDS/IPS monitoring server. The server captures all mirrored traffic and performs real-time anomaly detection using a deep learning-based Autoencoder. Attacker machines (Kali-Linux and Ubuntu-Desktop) simulate cyberattacks - such as TCP SYN Flood and ICMP Flood. The monitoring system visualizes alerts via a web-based dashboard and can dynamically block malicious hosts by issuing Telnet commands to network devices. Client machines (PC1 and PC2) serve as attack targets, providing a realistic environment for testing and validating the model. This flexible detection and response mechanism allows the network to continue functioning normally while effectively mitigating both internal and external threats.

❖ **Operation of the monitoring system (AI-IDS/IPS)**

On the AI-IDS/IPS monitoring server (Figure 4), the system provides a Web Dashboard interface developed using FastAPI, integrated with Tailwind CSS and WebSocket technology to display device information tables, alert statistics charts, and real-time updates. Administrators can monitor device statuses, review the history of detected anomalies, and view the response actions performed by the system.

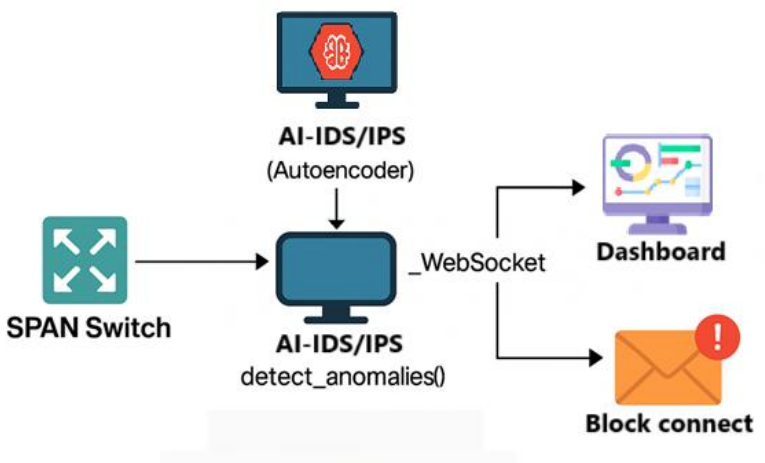


Figure 4. Operational model of the monitoring system (AI-IDS/IPS)

4. EXPERIMENTAL RESULTS AND DISCUSSION

4.1. System testing

4.1.1. Data collection and AI model training

The *scan_system* module collects network device data over a predefined period of 30 seconds (***Figure 5***), with estimated scanning durations detailed in ***Table 2***. The system determines device connection duration using a cache file (*device_cache.json*), which records

when each device is first detected and calculates its total time on the network. All device data is normalized and saved to *network_device_data.csv*.

This module enables the system not only to monitor real-time traffic but also to build a comprehensive dataset for more effective AI model training and anomaly detection.

```
(notebook-env) root@ainac:/home/admin2025/notebook-env/bin/AI-NAC# python3 scan_system.py
||==> Starting network monitoring (Press Ctrl+C to stop)
||==> [17:30:08] Scanning with Nmap...
||==> Sniffing packets on interface 'ens37' for 30s…
OK ! Saved 8 records to network_device_data.csv
```

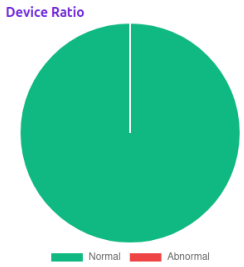
Figure 5. Device scanning and information collection interface

Estimated time for device scanning:

IP range (Subnet mask)	Number of IPs	Sequential scanning	Multithreaded scanning (scapy hoặc asyncio)
/30	4	~2s	<2s
/24	256	~1-2 minutes	~5-10 seconds
/22	1024	~8-10 minutes	~20-30 seconds
/16	65.536	>1 hour	~5-10 minutes

Table 2. Estimated time for device scanning

The statistical chart showing the ratio of normal/abnormal devices, along with a table displaying device information (IP address, MAC address, packet count, data volume, duration, and status), is illustrated in ***Figure 6*** on the Web Dashboard.



IP Address	MAC	Packets	Bytes	Time	Status
192.168.75.1	00:50:56:C0:00:08	16	2030	2025-08-05 16:28:17	NORMAL
192.168.75.129	00:0C:29:A3:3D:C7	29	3423	2025-08-05 16:28:17	NORMAL
192.168.75.130	00:0C:29:16:01:C2	0	0	2025-08-05 16:28:17	NORMAL
192.168.75.131	00:0C:29:81:26:6D	0	0	2025-08-05 16:28:17	NORMAL
192.168.75.133	00:0C:29:FB:84:B1	0	0	2025-08-05 16:28:17	NORMAL
192.168.75.140	UNKNOWN	0	0	2025-08-05 16:28:17	NORMAL
192.168.75.2	00:50:56:EF:87:58	33	12110	2025-08-05 16:28:17	NORMAL
192.168.75.254	00:50:56:EB:24:FE	0	0	2025-08-05 16:28:17	NORMAL

Figure 6. Statistical chart of normal and abnormal device status on the Dashboard

The AI-IDS/IPS web dashboard interface displays statistics including the total number of devices, the number of anomalous devices, and the last update time. It also presents a chart visualizing the reconstruction error of each device across specified error ranges, as illustrated in ***Figure 7***.

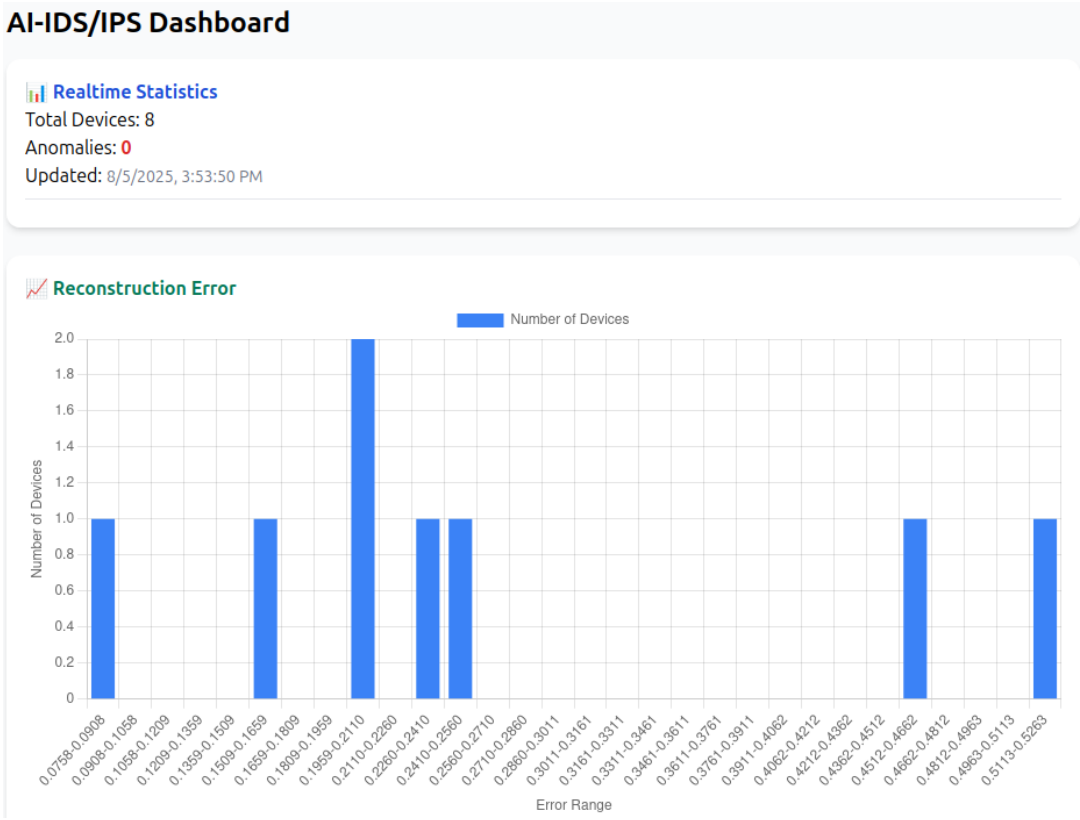


Figure 7. Device count statistics and reconstruction error chart

4.1.2. Attack experiment and response of the AI-IDS/IPS

During the performance evaluation of the AI-IDS/IPS system, real-world attack experiments were conducted to assess the model's capability to detect and respond to common network threats. Two representative types of attacks and their respective tools were chosen to represent widely used attack techniques, including:

- ICMP Flood
- TCP SYN Flood

❖ **Scenario 1: Denial of Service attack - ICMP Flood**

In this attack scenario, the Ubuntu-Desktop machine (IP 192.168.75.130) uses the *hping3* tool to perform a Denial of Service (DoS) attack by flooding the target device with a large number of ICMP Echo Request (ping) packets. The target machine is PC1, with IP address 192.168.75.11. This attack may cause disruption or even crash the victim system.

Figure 8 shows the interface where the command is executed:

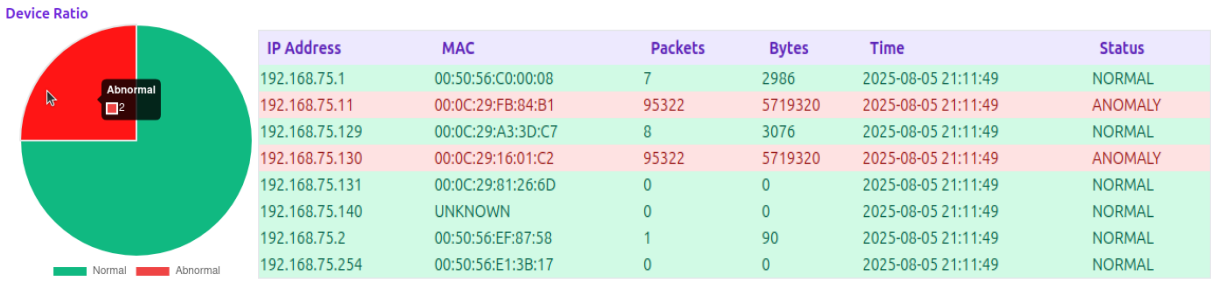
hping3 -1 --flood 192.168.75.11

```
root@admin2025:/home/admin2025# hping3 -1 --flood 192.168.75.11
HPING 192.168.75.11 (ens33 192.168.75.11): icmp mode set, 28 headers + 0 data by
tes
hping in flood mode, no replies will be shown
```

Figure 8. Command Line Interface on the Ubuntu-Desktop PC

Response of the AI-IDS/IPS System:

The monitoring system detected a sudden spike in ICMP packets from a single source within a short timeframe. The AI model recognized this as an anomaly, triggering an immediate alert on the Web Dashboard. As illustrated in ***Figure 9***, the dashboard shows one anomalous device with IP address 192.168.75.130 - the attacking machine - sending 95,322 packets totalling 5,719,320 bytes to the target machine.



IP Address	MAC	Packets	Bytes	Time	Status
192.168.75.1	00:50:56:C0:00:08	7	2986	2025-08-05 21:11:49	NORMAL
192.168.75.11	00:0C:29:FB:84:B1	95322	5719320	2025-08-05 21:11:49	ANOMALY
192.168.75.129	00:0C:29:A3:3D:C7	8	3076	2025-08-05 21:11:49	NORMAL
192.168.75.130	00:0C:29:16:01:C2	95322	5719320	2025-08-05 21:11:49	ANOMALY
192.168.75.131	00:0C:29:81:26:6D	0	0	2025-08-05 21:11:49	NORMAL
192.168.75.140	UNKNOWN	0	0	2025-08-05 21:11:49	NORMAL
192.168.75.2	00:50:56:EF:87:58	1	90	2025-08-05 21:11:49	NORMAL
192.168.75.254	00:50:56:E1:3B:17	0	0	2025-08-05 21:11:49	NORMAL

Figure 9. Anomaly detection and alerts displayed on the AI-IDS/IPS Dashboard

❖ **Scenario 2: Denial-of-Service Attack - TCP SYN Flood and ICMP Flood**

In this scenario, a coordinated attack is simulated from two machines - Kali-Linux (IP 192.168.75.131) and Ubuntu-Desktop (IP 192.168.75.130) - simultaneously targeting the victim machine, PC1 (IP 192.168.75.11). The Kali-Linux machine performs a ICMP Flood operation using *hping3* to overload the CPU and bandwidth of PC1, while the Ubuntu-Desktop machine conducts a TCP SYN Flood attack using *hping3* against the same target. ***Figure 10*** displays the command execution interface and the results of the scan:

hping3 -1 --flood 192.168.75.11

```
┌──(root㉿kali)-[~]
└─# hping3 -1 --flood 192.168.75.11
HPING 192.168.75.11 (eth0 192.168.75.11): icmp mode set, 28 headers + 0 data
bytes
hping in flood mode, no replies will be shown
```

Figure 10. Probing and Information Gathering on PC1

Figure 11 shows the command execution interface from the Ubuntu-Desktop machine:

hping3 -S -p 80 --flood 192.168.75.11

```
root@admin2025:/home/admin2025# hping3 -S -p 80 --flood 192.168.75.11
HPING 192.168.75.11 (ens33 192.168.75.11): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 11. Command Line Interface on the Ubuntu-Desktop machine

Response of the AI-IDS/IPS System:

In **Figures 12** and **13**, the system detected a sudden spike in TCP SYN connection frequency across multiple ports, flagging it as an anomaly. An alert was triggered with the logged IP address of the suspicious device, which is displayed on the Web dashboard. The information table lists two anomalous devices, accompanied by a reconstruction error chart of all monitored devices.

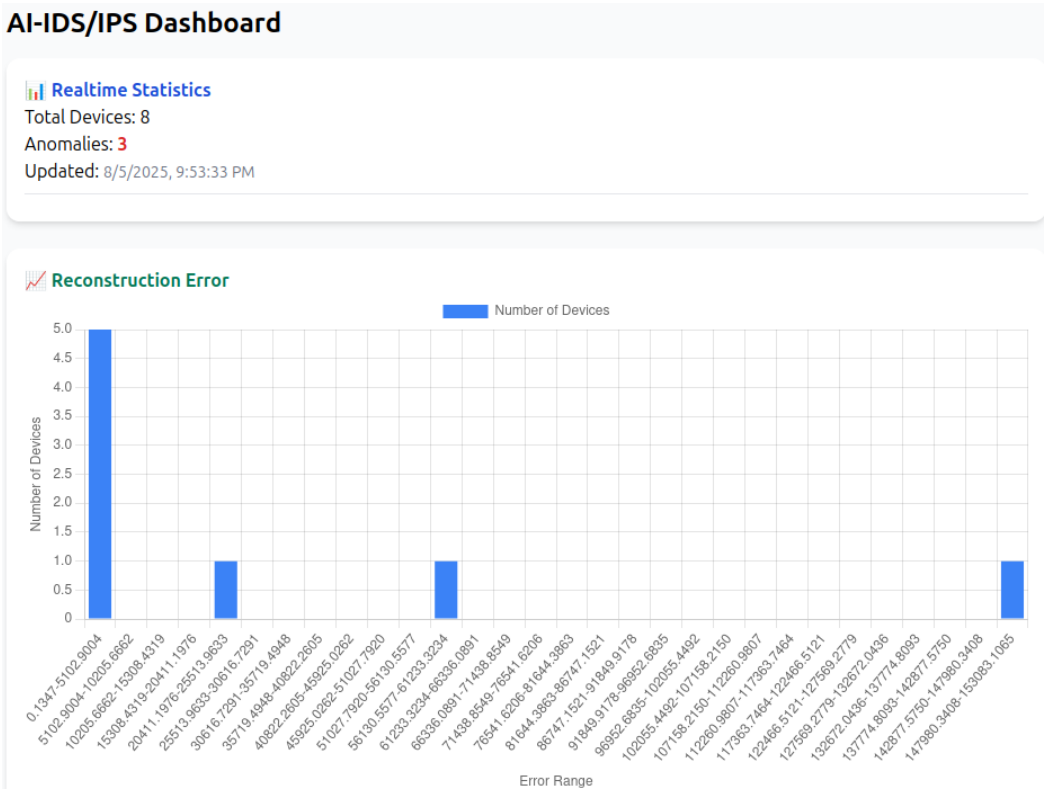
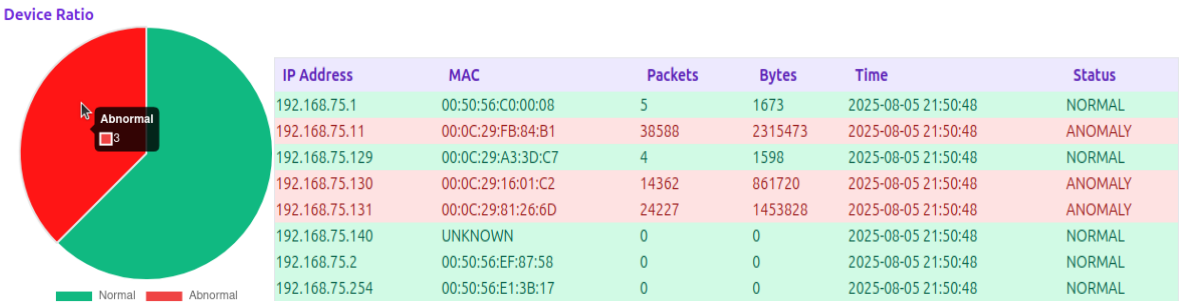


Figure 12. Statistics of detected anomalous devices and reconstruction error chart



IP Address	MAC	Packets	Bytes	Time	Status
192.168.75.1	00:50:56:C0:00:08	5	1673	2025-08-05 21:50:48	NORMAL
192.168.75.11	00:0C:29:FB:84:B1	38588	2315473	2025-08-05 21:50:48	ANOMALY
192.168.75.129	00:0C:29:A3:3D:C7	4	1598	2025-08-05 21:50:48	NORMAL
192.168.75.130	00:0C:29:16:01:C2	14362	861720	2025-08-05 21:50:48	ANOMALY
192.168.75.131	00:0C:29:81:26:6D	24227	1453828	2025-08-05 21:50:48	ANOMALY
192.168.75.140	UNKNOWN	0	0	2025-08-05 21:50:48	NORMAL
192.168.75.2	00:50:56:EF:87:58	0	0	2025-08-05 21:50:48	NORMAL
192.168.75.254	00:50:56:E1:3B:17	0	0	2025-08-05 21:50:48	NORMAL

Figure 13. AI-IDS/IPS anomaly detection and alerting on the Dashboard

4.1.3. Evaluation of anomaly detection and AI response effectiveness

The performance evaluation of the AI-IDS/IPS system is divided into several key stages: network scanning latency (scan_system), anomaly detection latency (using Autoencoder and MLP models), and the alert display latency via WebSocket on the dashboard. Measurements were conducted using the time library to record the start and end times of the scan_system() function. Additionally, timestamps at each processing phase - including packet reception, anomaly detection, alert transmission via WebSocket, and device control response via Telnet - were logged through the system's logging mechanism. The Dashboard interface measures real-time display latency using the onmessage() event of WebSocket and the timestamp at the DOM (Document Object Model) rendering time on the frontend. The measured results are aggregated and displayed directly on the dashboard to support real-time performance monitoring.

Specifically, as shown in ***Figure 14****, system performance monitoring is presented with two axes: latency (in seconds) and time intervals measured every 5 seconds. Through experimental evaluation, the average time from the start of the attack to the completion of the system's response ranges between 25 and 35 seconds. The anomaly detection and alerting process takes approximately 10-15 seconds. Following detection, the system sends an alert through WebSocket to the dashboard and initiates a Telnet connection from the AI-IDS/IPS device to the switch to block packets, which takes another 10-15 seconds.*

Therefore, the AI-IDS/IPS system design (shown in ***Figure 3****) effectively demonstrates the capability to detect anomalies and respond effectively within a short time frame.*

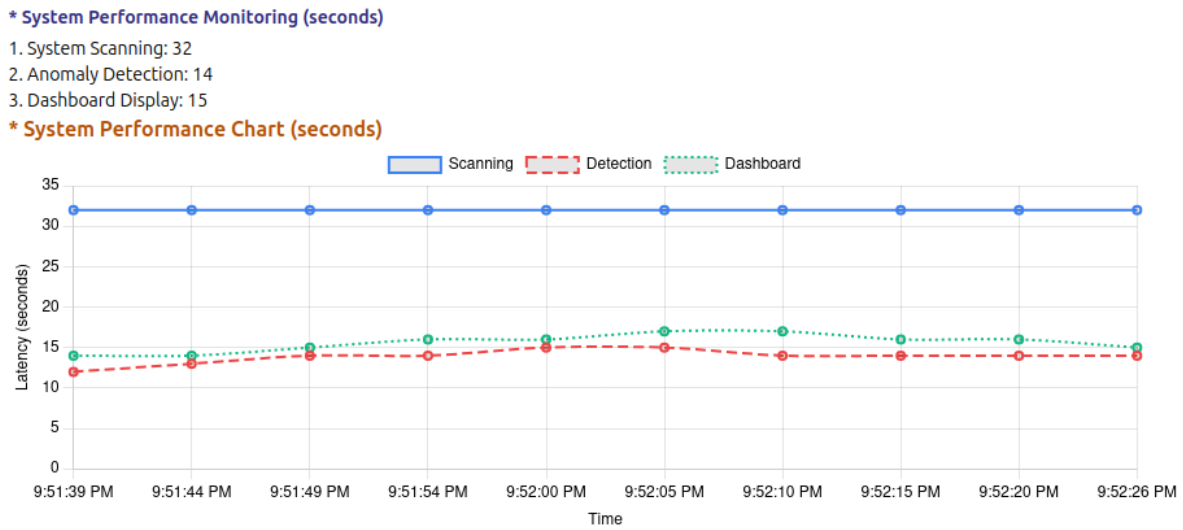


Figure 14. Performance monitoring chart of system alerts on the Dashboard

However, in large-scale and wide-area network models with many devices, latency increases significantly in subnet ranges of /22 (1,024 IP addresses) and /16 (65,536 IP

addresses) if the scanning scope is not subdivided. Additionally, AI inference on the GPU needs to be optimized, as it may cause CPU resource bottlenecks when handling multiple concurrent connections and tasks.

To reduce latency in large-scale network systems, several solutions should be implemented: subdividing subnets to reduce the processing load per segment; increasing the number of AI-IDS/IPS monitoring units deployed in local areas; and optimizing GPU inference processes to offload the CPU.

4.2 Quantitative evaluation of model performance

Autoencoder is an unsupervised deep learning model used to extract anomalous features from network data via reconstruction error. By applying a predefined threshold to this score, each session is automatically assigned a binary pseudo label (0: normal, 1: anomalous). These pseudo labels are then used to train a supervised classification model, specifically a Multi Layer Perceptron (MLP), which aims to approximate the anomaly definition induced by the Autoencoder.

The test dataset consists of 12,438 network sessions collected over a period of three days, covering three traffic types: Ping, ICMP Flood, and Port Scan. This dataset was assigned binary pseudo-labels (0: normal, 1: anomalous) in a preprocessing stage that applied an Autoencoder-based reconstruction-error threshold to each session. After training the MLP classifier on 80% of the dataset, the model was applied to the remaining 20% to predict output labels. The MLP model outputs a probability (ranging from 0 to 1) representing the likelihood that a session is anomalous. This probability is then thresholded at 0.5 to assign binary labels (if the output > 0.5, the session is labeled as anomalous). The corresponding command is: *y_pred = (mlp.predict(X_test) > 0.5).astype(int)*.

Next, the confusion matrix is computed using the confusion_matrix function from the sklearn.metrics library, as shown below:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
TN, FP, FN, TP = cm.ravel()
```

The confusion matrix yields four key values - TP, FP, FN, and TN - which are presented in ***Table 3***:

- TP (True Positive) = 1,812: Number of anomalous sessions correctly detected by the model.

- FP (False Positive) = 284: Number of normal sessions incorrectly identified as anomalous.
- FN (False Negative) = 188: Number of anomalous sessions missed by the model (predicted as normal).
- TN (True Negative) = 10,154: Number of normal sessions correctly identified by the model.

Performance evaluation metrics:

Metric	Meaning	Formula	Value
Precision	Precision of anomaly detection: The ratio of correctly predicted anomalies to the total number of predicted anomalies, reflecting the false alarm rate.	$\frac{TP}{TP + FP}$	0,865
Recall	Recall (Detection Rate): The proportion of correctly detected anomalies out of all actual anomalies, reflecting the model's ability to avoid missing attacks.	$\frac{TP}{TP + FN}$	0,906
Accuracy	Overall accuracy: The ratio of correctly classified sessions to the total number of sessions.	$\frac{TP + TN}{TP + TN + FP + FN}$	0,962
F1-score	Balanced metric (F1 Score): The harmonic mean of Precision and Recall, balancing the trade-off between false alarms and missed attacks.	$2 \times \frac{Precision \times Recall}{Precision + Recall}$	0,885

Table 3. Performance evaluation metrics

The results indicate that the model achieved a high accuracy of 96.2% and a high Recall of 90.6%, effectively meeting the requirements for early anomaly detection.

The obtained Precision, Recall, Accuracy, and F1-score thus reflect how consistently the MLP reproduces the Autoencoder-based anomaly definition over the collected sessions, suggesting its potential suitability for near real-time network monitoring, particularly for DoS flood scenarios tested in the simulated LAN environment.

5. CONCLUSION

This research proposes an intelligent AI-IDS/IPS model that utilizes an Autoencoder - an unsupervised deep learning architecture - for real-time anomaly detection in network traffic. Through experimental evaluations involving common attack types such as TCP SYN Flood and

ICMP Flood, the system demonstrated effective capabilities in detecting anomalies and responding promptly by logging events, issuing alerts, and blocking traffic from malicious sources.

In terms of performance, the system exhibits a noticeable increase in processing latency within large-scale network environments, particularly when operating across wide subnets or under high device density conditions. Additionally, since the detection mechanism primarily relies on statistical features of network packets, it may face difficulties in identifying stealthy or long-duration attacks (commonly referred to as low and slow attacks).

Future research will focus on optimizing the system's processing infrastructure - especially by leveraging AI inference on GPUs to reduce the computational load on CPUs. Enhancements will also include the integration of Reinforcement Learning techniques and User Behavior Analytics (UBA) to improve the detection of high-level anomalous behaviors. Furthermore, plans include expanding the training dataset to accommodate additional attack patterns such as ARP Spoofing, Brute Force attacks and HTTP Floods, as well as deploying the AI-IDS/IPS system under edge or hierarchical architectures to ensure scalability while maintaining high performance and responsiveness.

With these improvements, the system is expected to achieve greater accuracy, enhanced adaptability, and higher reliability, ultimately contributing to the development of a secure, proactive, and resilient network environment capable of addressing modern cybersecurity challenges.

REFERENCES

[1] K. An, "Hơn 659.000 vụ tấn công mạng cơ quan, doanh nghiệp năm 2024," *Lao Động*, 2024. [Online]. Available: https://laodong.vn/ban-doc/hon-659000-vu-tan-cong-mang-co-quan-doanh-nghiep-nam-2024-1439390.ldo

[2] A. L. Giri and S. Annamalai, "Intrusion detection system for local networks - A review study," in *Proc. 2nd Int. Conf. on Advances in Computing, Innovation and Technology in Engineering (ICACITE)*, pp. 1388–1391, 2022.

[3] A. S. Ashoor and S. D. Gore, "Difference between Intrusion Detection System (IDS) and Intrusion Prevention System (IPS)," *Communications in Computer and Information Science*, vol. 151, pp. 1–6, Springer, India, Jul. 2011.

[4] M. M. Issa, M. Aljanabi, and H. M. Muhialdeen, "Systematic literature review on intrusion detection systems: Research trends, algorithms, methods, datasets, and limitations," *Journal of Intelligent Systems*, vol. 33, no. 1, pp. 15–26, 2024.

[5] H.-Y. Kwon, T. Kim, and M.-K. Lee, "Advanced intrusion detection combining signature-based and behavior-based detection methods," *Electronics*, vol. 11, no. 6, pp. 2–4, 2022.

[6] F. Wang and L. Yao, "A deep learning approach for intrusion detection system with reduced false positives," *Journal of Network and Computer Applications*, vol. 155, pp. 2–10, 2020.

[7] J. F. Canola Garcia and G. E. Taborda Blandon, "A deep learning-based intrusion detection and prevention system for detecting and preventing denial-of-service attacks," *IEEE Access*, vol. 10, pp. 83050–83055, 2022.

[8] S. Kumar, S. Gupta, and S. Arora, "Research trends in network-based intrusion detection systems: A review," *IEEE Access*, vol. 9, pp. 157761–157774, 2021.

[9] S. S. Raghavan, "A comprehensive study of Artificial Intelligence applications in Intrusion Detection and Prevention," *Int. J. of AI Research and Development (IJAIRD)*, vol. 3, no. 1, pp. 16–37, Jan.–Jun. 2025.

[10] V. N. Kumar, D. Siri, and A. Badhoutiya, "Anomaly detection in provenance data using autoencoder network," in *Proc. IEEE Conf. on Intelligent Systems*, 2024. ISBN: 979-8-3303-6810-9.

[11] H. Torabi, S. L. Mirtaheri, and S. Greco, "Practical autoencoder-based anomaly detection by using vector reconstruction error," *Cybersecurity*, vol. 6, art. 1, 2023.

[12] S. Ang, S. Huy, and M. Janarthanan, "Utilizing IDS and IPS to Improve Cybersecurity Monitoring Process," *Journal of Cyber Security and Risk Auditing*, vol. 2025, no. 3, pp. 77–88, 2025.

[13] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artificial Intelligence Review*, vol. 55, no. 4, pp. 470–510, 2022.

[14] Asif Ahmed Neloy, Maxime Turgeon, "A comprehensive study of auto-encoders for anomaly detection: Efficiency and trade-offs", *Machine Learning with Applications 17*, 100572, July 2024

[15] S. Raschka and V. Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn and TensorFlow 2*, 3rd ed., Birmingham, U.K.: Packt Publishing, 2019.